REAL TIME VEHICLE COUNTING USING YOLO-TINY

Camelia Florea
Communications Department, UTCN
Camelia.Florea@com.utcluj.ro

WHO AM I?

- Currently, I am an Assistant Professor
  - at the Technical University of Cluj-Napoca,
  - in the Department of Communication,
  - Faculty of Electronics, Telecommunications and Information Technology.

WHO AM I?

- Research & teaching subjects:
  - signal/image/video processing and analysis,
  - images and video sequences compression,
  - compressed domain data processing,
  - machine learning,
  - deep learning,
  - multimedia technologies.

REAL-TIME VEHICLE DETECTION AND TRACKING

- Real-time vehicle detection, tracking and counting from surveillance cameras is a main part for many applications in smart cities.

- Usually, this task encounters some problems in practice,
  - like the lack of real-time processing of the videos
  - or the errors in ***detection and/or tracking***.

- In this presentation - an approach for real time vehicle counting
  - by using **Tiny YOLO for detection** and **fast motion estimation for tracking**.

- The application is running in Ubuntu
  - on a PC with GPU processing,
  - and on a low-budget devices, as Jetson Nano.

TRAFFIC MANAGEMENT SYSTEMS

- Real time traffic management systems,
  having vehicle counting as an important part,
    - has become popular recently due to
        - the availability of low-cost surveillance cameras,
        - the powerful mobile devices for video analysis
        - and the cloud infrastructure.

YOLO ("YOU ONLY LOOK ONCE“)

- YOLO is
  - a well-known Deep Neural Network
  - and powerful object detection algorithm

  with real-time performance on a computer with GPU (Graphics Processing Unit).
- YOLO has been trained using COCO (Common Objects in Context) dataset and can predict 80 classes such as car, bus, truck, motorbike, person, animals, food, etc.
- In presented implementation we consider just 3 classes: car, truck and bus, and we treat them as vehicles.
- The presence of a graphic card is a real demand to increase the speed of object detection using the pre-trained YOLO models.

YOLO PRE-TRAINED MODELS

- Due to the ***real time demands*** of our application
  - we chose to use YOLOv3-tiny

    \- assure the ***proper balance between accuracy and speed***.
- YOLOv3-spp is much more accurate but too slow for real time processing, even on GPU.
- We solve the accuracy problem of YOLOv3-tiny
  - by the fact that every vehicle is for sure detected in at least one of the frames (passing through the active ROI, since we use videos in our application).
  - once we detect a vehicle in the current frame, we track it
- We track the vehicle by checking
  - if it appears detected in the next frames, or,
  - if not detected, we estimate its position in the frame taking in account its last positions and ***the velocity and direction of movement***.

APPLICATION FLOWCHART

- In our implementation we consider just 3 classes: car, truck and bus, and we treat them as vehicles.
- The version YOLO-Tiny predicts bounding boxes using anchor boxes and multi-label classification,
  - therefore, for the same object we can have different bounding boxes detected,
  - or we can have the same object classified in two different classes (with different scores).

  ⇒If in the current frame a vehicle is detected twice, we filter the resulted bounding boxes to obtain unique identified vehicle.

We track the vehicle by checking its last positions and ***the velocity and direction of movement***.

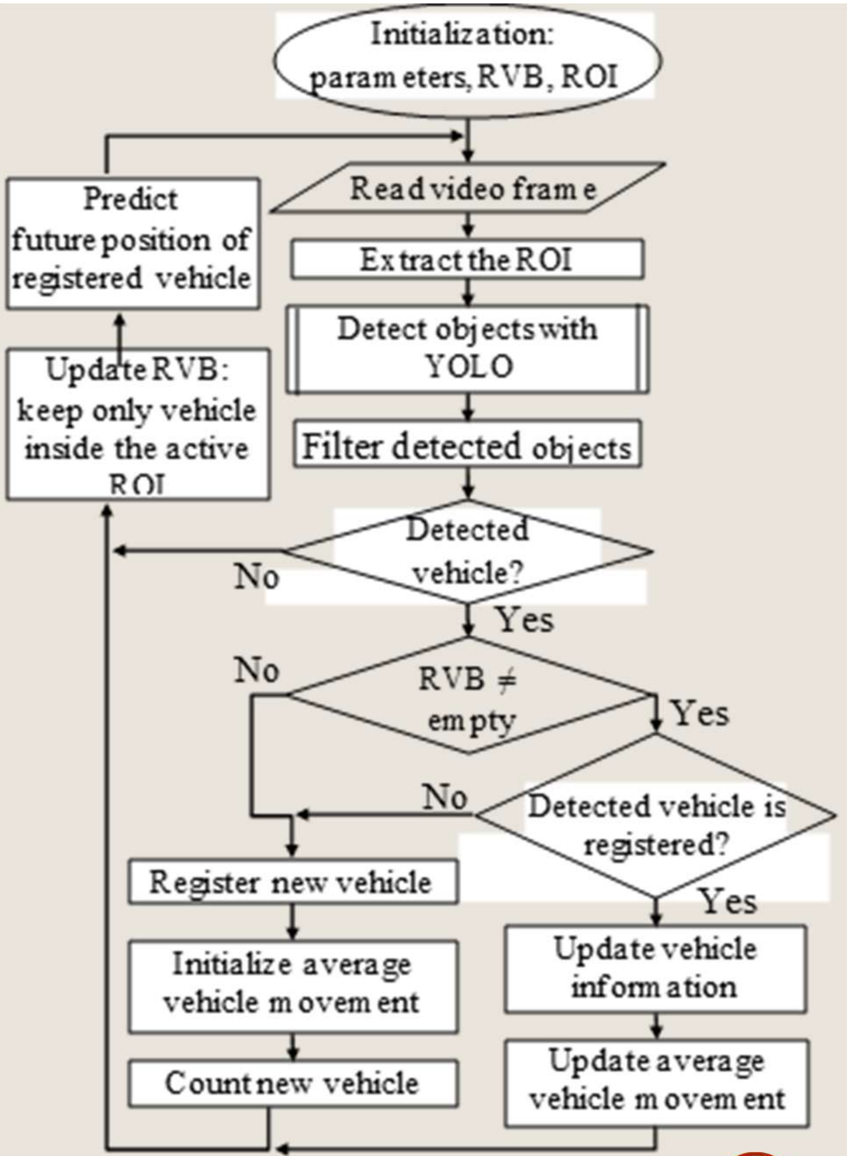
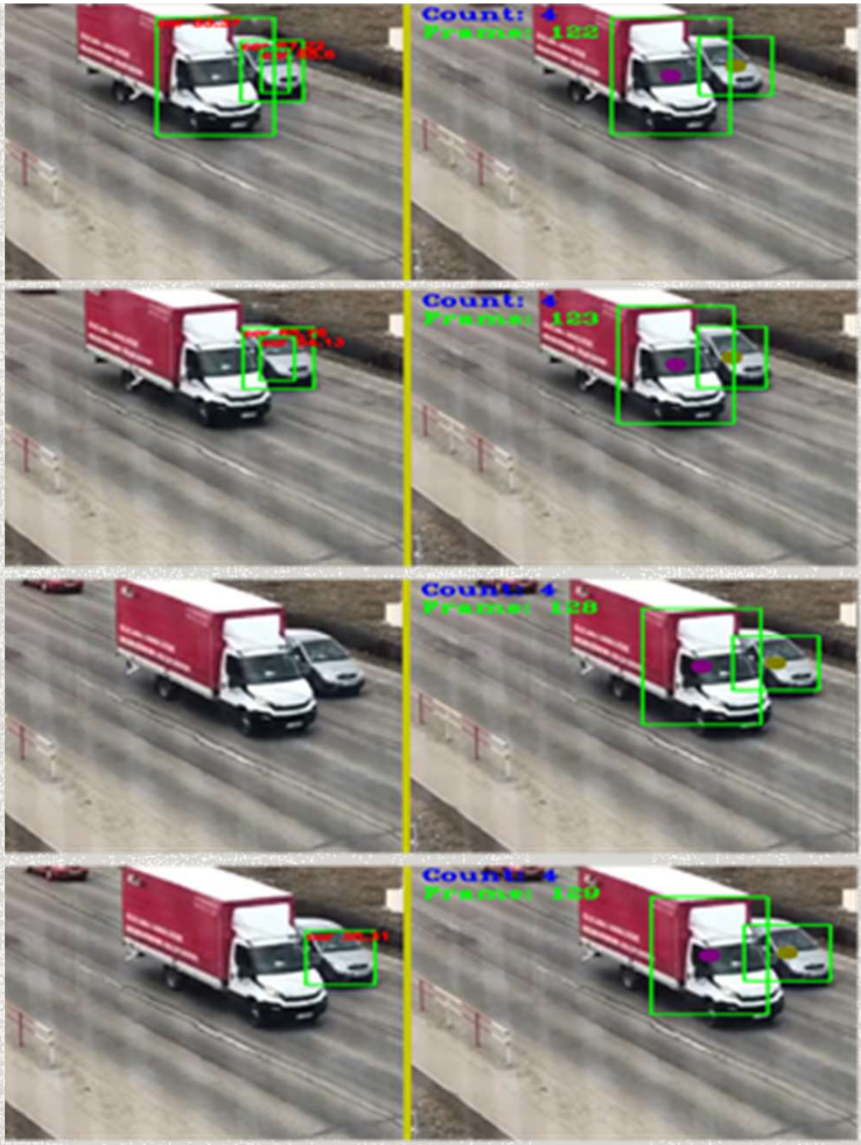


Fig. 1. Application flowchart.

EXPERIMENTAL RESULTS

- We tested our implementation with several movies containing real traffic videos.
  - later also on IP camera (real time streaming)
- Our application is developed
  - in Python on a computer running Ubuntu OS
  - uses GPU on the GeForce GTX 950M graphic card (650 CUDA core, compute capability 5.0).
  - the average speed was 33.5 FPS, qualifying our implementation for real-time operation
- The presence of the graphic card is a real demand to substantially increase the speed of operation.



. 1. A collection of four screenshots, illustrating the d cs of diction, tracking and vehicle counting for the first test movie.

EXPERIMENTAL RESULTS

- There are three vehicles in the scene, all of them with double detection (see the left window of the image), but with uniquely identification after the filtering operation (right window).
- They are correctly registered (different color for their graphical identifier), tracked and counted.
- The total number of unique counted vehicles is now 12.

Fig. 1. Two screenshots with the results for the first testmovie, frame 433 and frame 454

EXPERIMENTAL RESULTS

- A second test movie type is considered
  - the ROI is defined so that both directions of travel are contained inside.
  - A single counter is used, to count all unique vehicle, regardless their movement direction.
- In the Figure there are four vehicles, three of them going from left to right, and one going from right to left.
  - All vehicles are detected by YOLO.
  - In the frame 415 (top image in Fig.), after the filtering operation only three of them remains.
    - This happens because the bounding boxes for two cars (marked with a blue circle) are superimposed, so the IoU (Intersection over Union) method eliminates the smallest bounding box.
    - As a consequence, only the car marked with a red dot (left window of the image) is counted, so at that moment there are 25 vehicles counted.
  - Anyway, the small red car is not definitively lost, because two frames later (frame 417) it is correctly processed and counted, the counter indicating now 26 vehicles.
  - All the other vehicles are correctly tracked and counted during their transit to the ROI.

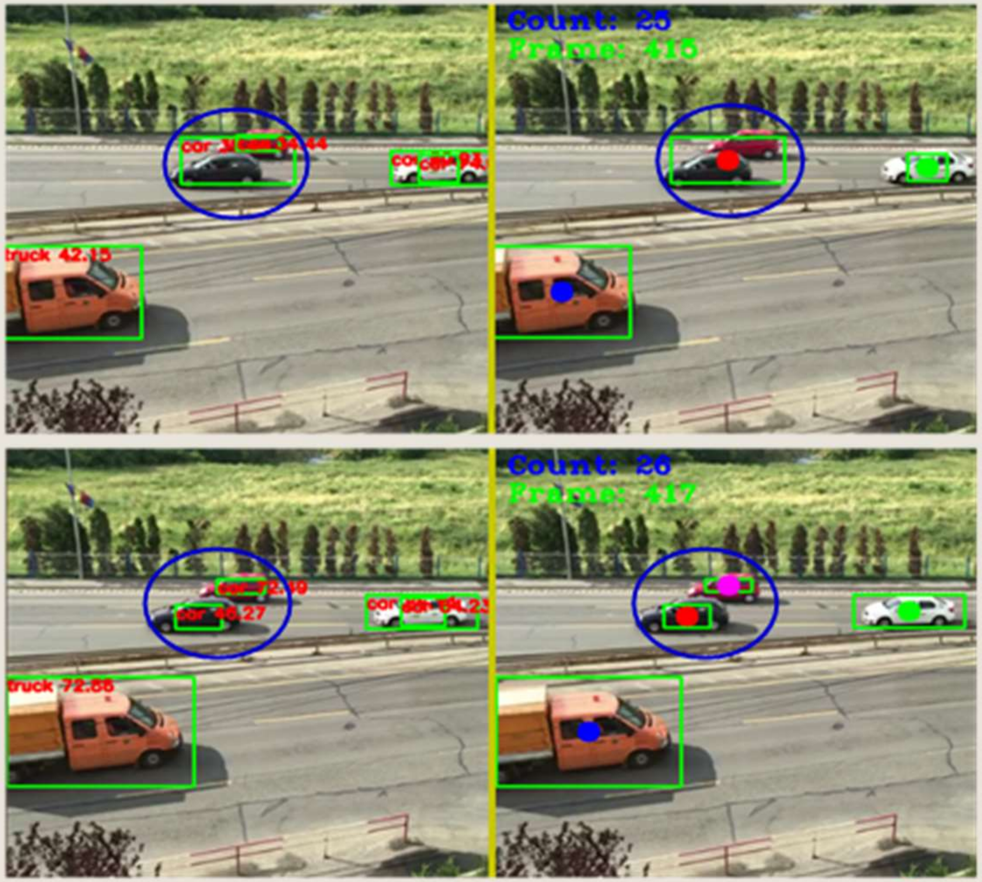


Fig. 1. Two screenshots with the results for the second test movie.

RETRAIN YOLO ON YOUR DATASET

GPU	NVIDIA GeForce GTX 780 TI	2,7 s
CPU	Intel Xeon CPU E5-1620, 3,60 GHz	214 s

```
car
truck
bus
```

```
# Training
batch=64
subdivisions=32
width=320
height=320
```

```
[convolutional]
size=1
stride=1
pad=1
filters=24
activation=linear
```

```
[yolo]
mask = 3,4,5
anchors = 56,31,
classes=3
```

```
classes= 3
train  = data/train.txt
names = data/obj.names
backup = backup/
```

filters=(classes+5)×3

JETSON NANO DEVELOPER KIT

https://developer.nvidia.com/embedded/jetson-nano-developer-kit

- It is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing

- It's just 89 USD

- Running the application on JETSON Nano
  - 14 FPS!!! – but good enough for vehicle counting


car
110
107
108
105
Frame: 784
Count: 110


car
110
111
107
108
Frame: 789
Count: 111


car
114
111
Frame: 806
Count: 114

Conclusion

- The vehicle detection uses the YOLOv3-tiny model that assures the best trade-off between speed and accuracy.
- A robust tracking mechanism, based on next position prediction is implemented, for correct vehicle counting.
- We tested our system using
  - few test movies with real traffic.
  - Real time stream using IP Camera
- The system operation is correct both for one way traffic, and two way traffic.
- A vehicle is counted only once,
  - even if it is detected in multiple frames,
  - or if there is a lack of detection in some intermediate frames, and then the vehicle is detected again.
- Details in the article:
  - Gabriel Oltean, Camelia Florea, Radu Orghidan, Victor Oltean, „***Towards Real Time Vehicle Counting Using YOLO-Tiny and Fast Motion Estimation***", 2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME), Cluj-Napoca, 23-26 October 2019.